Discrete Applied Mathematics 10 (1985) 125–137
North-Holland

A HEURISTIC APPROACH TO HARD CONSTRAINED SHORTEST PATH PROBLEMS

Celso C. RIBEIRO
Department of Electrical Engineering, Catholic University of Rio de Janeiro, Gavea, Caixa Postal 38063, Rio de Janeiro 22452, Brazil

Michel MINOUX
Department of Applied Mathematics, National Center for Telecommunication Studies, 38 rue du Général Leclerc, Issy-Les-Moulineaux 92131, France

Received 15 January 1984

Constrained shortest path problems have many applications in areas like network routing, investments planning and project evaluation as well as in some classical combinatorial problems with high duality gaps where even obtaining feasible solutions is a difficult task in general.

We present in this paper a systematic method for obtaining good feasible solutions to hard (doubly constrained) shortest path problems. The algorithm is based essentially on the concept of efficient solutions which can be obtained via parametric shortest path calculations. The computational results obtained show that the approach proposed here leads to optimal or very good near optimal solutions for all the problems studied.

From a theoretical point of view, the most important contribution of the paper is the statement of a pseudopolynomial algorithm for generating the efficient solutions and, more generally, for solving the parametric shortest path problem.

1. Introduction and problem statement

We study here the shortest path problem in a graph, with an additional constraint in the form of a double sided inequality. Let $G=[N, U]$ be an oriented connected graph where $N=\{1,2,\ldots,n\}$ is the set of nodes and $U=\{1,2,\ldots,m\}$ is the set of arcs. We denote by $u=(i,j)$ the arc where i is the initial node and j the terminal node. To each arc $u\in U$ we attach two integers:

- c_u, the cost incurred by traversing arc u, and
- r_u, the quantity of some resource used when traversing arc u.

The c_u and r_u can take on any values provided that G contains no negative circuit with respect to c_u. Given an upper bound $\bar{r}$ and a lower bound $\underline{r}$ on the total amount of the given resource to be used, we address the problem of finding a minimum cost elementary path P between two specified nodes s (origin) and t (destination) such that the additional resource constraint (double sided inequality) $\underline{r}\leq\sum_{u\in P} r_u\leq\bar{r}$ is satisfied.

To any elementary path P between s and t in G we associate its characteristic vector $x=(x_1,x_2,\ldots,x_m)\in\{0,1\}^m$ such that $x_u=1$ if and only if $u\in P$. Let G' be a

subgraph of G. Then, for any pair of nodes i and j $(i \neq j)$ of this subgraph G', we denote by $X_{ij}(G')$ the set of all characteristic vectors of elementary i–j paths in G'. Let

$$c \cdot x = \sum_{u \in U} c_u x_u \quad \text{and} \quad r \cdot x = \sum_{u \in U} r_u x_u.$$

The doubly constrained shortest path problem may be formulated as:

$$\begin{array}{lll} \text{[DCSP]} & \text{minimize} & c \cdot x, \\ & \text{subject to} & \underline{r} \leq r \cdot x \leq \bar{r}, \\ & & x \in X_{st}(G). \end{array}$$

It is noticed that the above formulation includes, as a special case, the shortest path problem with one equality constraint (take $\underline{r} = \bar{r}$).

An important application of this formulation to the resolution of hard knapsack problems is studied, among other applications, in [13] and in [12]. In these references we show how to transform equality constrained knapsack problems into doubly constrained shortest path problems where the values $\underline{r}$ and $\bar{r}$ are usually different.

Previous work on constrained shortest path problems has been carried out along two main directions:

- The study of the shortest path problem with one single inequality constraint, later on referred to as [SCSP] (see Joksch [7], Minoux [10], and Handler & Zang [6]). A good survey of its applications can be found in Minoux [10], including scheduling on probabilistic PERT networks, optimal network expansion subject to a budgetary constraint, shortest path on probabilistic graphs and routing in communication networks.
- The study of time constrained shortest path problems arising in the context of railroad networks management (see Halpern & Priess [5]) and routing problems with time windows in transportation networks (see Desrosiers, Soumis & Desrochers [2]).

The doubly constrained shortest path problem [DCSP] (with the requirement that the optimal path should be elementary) does not seem to have been studied so far. In an early paper by Saigal [14] it is shown that the equality constrained problem can be converted into a cardinality constrained shortest path problem on an associated graph; however, non-elementary paths can be obtained as solutions.

Problem [DCSP] is NP-complete (see Ribeiro [13]). Problem [SCSP] is also NP-complete, as it was shown by N. Meggido (see Garey & Johnson [4]). One argument which explains why [DCSP] is usually much harder than [SCSP] is that it includes as a special case the knapsack problem with an equality constraint. In particular, it may be easily realized that even finding feasible solutions to [DCSP] is a difficult task in itself. The purpose of this paper is actually to devise a procedure for generating good feasible solutions to the doubly constrained shortest path problem.

The approach presented here relies on the concept of *efficient solutions,* and on a fast algorithm for generating the efficient solutions via parametric shortest

path computations. From a theoretical point of view, an important contribution of the paper is the design of a pseudopolynomial algorithm for solving the general parametric shortest path problem. This extends a previous result obtained by Karp & Orlin [8] on a special case of parametric shortest paths.

2. A heuristic approach based on the concept of efficient solutions

Basic to the approach presented here is a partition $\{N_1, N_f, N_2\}$ of the set N of nodes of G such that $s \in N_1$ and $t \in N_2$. Let G_1 (resp. G_2) be the subgraph of G induced by $N_1 \cup N_f$ (resp. $N_f \cup N_2$) with arc set U_1 (resp. U_2). We call *partial solution* of [DCSP] with respect to the partition $\{N_1, N_f, N_2\}$ the characteristic vector $y(j)$ associated to a path from s to $j \in N_f$ which takes only arcs of U_1. We call *final solution* of [DCSP] with respect to the partition $\{N_1, N_f, N_2\}$ the characteristic vector $z(j)$ associated to a path from $j \in N_f$ to t which takes only arcs of U_2.

We define the cost of the partial solution having $y(j)$ as characteristic vector as $c(y(j)) = \sum_{u \in U_1} c_u y_u(j)$, and, analogously, we define its resource consumption as $r(y(j)) = \sum_{u \in U_1} r_u y_u(j)$.

A partial solution $y(j)$ combined with a final solution $z(j)$ gives rise to a path from s to t in G which will be denoted $y(j) \oplus z(j)$. The approximate algorithm to be described here is based on a systematic way of *generating* a subset of partial solutions (later on referred to as *efficient solutions*) and combining them with all possible final solutions (obtained by enumeration), so as to produce good feasible solutions of [DCSP]. Thus, in general, the partition $\{N_1, N_f, N_2\}$ will be chosen in such a way that the cardinalities of the sets $X_{jt}(G_2)$ of final solutions are not too large. Also, in most cases, N_f will be taken as an s–t disconnecting set of vertices (i.e., a subset of vertices which are met by any s–t path in G).

In order to introduce the concept of efficient solution, let us consider a mathematical programming problem in the general form:

$$\text{[Q]} \qquad f(x^*) = \operatorname{Min} f(x)$$
$$\text{subject to} \quad g_i(x) = 0 \quad (i = 1, 2, \ldots, m)$$
$$x \in X \subset \mathbb{R}^n,$$

where f and g_i $(i = 1, 2, \ldots, m)$ are real convex functions on $\mathbb{R}^n$, and X is a subset of $\mathbb{R}^n$ (for our purpose here, it is enough to assume that X is finite). To each constraint $g_i(x) = 0$ we attach a multiplier λ_i (unconstrained in sign), which can be interpreted as a penality associated with the violation of the constraint, and we define the *lagrangian function* as

$$L(x, \lambda) = f(x) + \sum_{i=1}^{m} \lambda_i g_i(x) = f(x) + \lambda \cdot g(x).$$

For any given set of multipliers λ, the so-called *dual function* value $w(\lambda)$ is then obtained by solving

$$[Q(\lambda)] \qquad w(\lambda) = \text{Min } L(x, \lambda)$$
$$\text{subject to} \quad x \in X,$$

and the following unconstrained maximization problem

$$[DQ] \qquad w(\lambda^*) = \max_{\lambda \in \mathbb{R}^m} w(\lambda)$$

is called the *dual program* corresponding to [Q].

Duality has been known for a long time as a very fundamental tool in mathematical programming from both theoretical and practical points of view (for a fairly complete survey see e.g., Minoux [11]). The main result which will be extensively used in the rest of the paper is the following:

Theorem 1 (Everett [3]). *Take* $\bar{\lambda} \in \mathbb{R}^m$ *and let* $\bar{x}$ *be the optimal solution of* $[Q(\bar{\lambda})]$. *Then* $\bar{x}$ *is the optimal solution of the (perturbed) problem*

$$[PQ] \qquad \text{Min } f(x)$$
$$\text{subject to} \quad g_i(x) = g_i(\bar{x}) \quad (i = 1, 2, \ldots, m)$$
$$x \in X,$$

i.e., the problem deduced from [Q] *by changing the right-hand sides of the constraints to* $g_i(\bar{x})$ $(i = 1, 2, \ldots, m)$.

Proof. See Everett [3]. □

In view of Theorem 1 above, we shall call an *efficient solution* of [Q] any $\bar{x} \in X$ for which there exists $\bar{\lambda} \in \mathbb{R}^m$ such that $\bar{x}$ is an optimal solution of $[Q(\bar{\lambda})]$. The importance of efficient solutions thus lies on their being the optimal solutions of some *perturbed problems* closely related to the original problem.

Considering the doubly constrained shortest path problem [DCSP], it seems that *efficient partial solutions* should be considered as good candidates to be combined with final solutions, in order to get approximate solutions of [DCSP]. Thus, for any $j \in N_f$, we denote by E_j the set of all efficient solutions of the problem

$$[P_j] \qquad \text{Min } c(y(j))$$
$$\text{subject to} \quad r(y(j)) = 0,$$
$$y(j) \in X_{sj}(G_1).$$

For $\lambda \in \mathbb{R}$, let $L_j(y, \lambda) = c(y) + \lambda r(y)$ be the lagrangian function of problem $[P_j]$. All the efficient solutions of $[P_j]$ can be obtained by solving

$$[P_j(\lambda)] \qquad L_j(\bar{y}(j, \lambda), \lambda) = \text{Min } L_j(y, \lambda)$$
$$\text{subject to} \quad y \in X_{sj}(G_1)$$

for all $\lambda \in [\lambda_{\min}, \lambda_{\max}]$, where $\lambda_{\min}$ (resp. $\lambda_{\max}$) is the smallest (resp. the greatest)

value of λ for which G_1 has no negative length circuits with respect to the lengths $c_u + \lambda r_u$ on the arcs $u \in U_1$.

A geometrical interpretation of the efficient solutions can be given by considering the piecewise linear concave function of one variable defined by $w_j(\lambda) = L_j(\bar{y}(j,\lambda),\lambda)$. The efficient solutions which correspond to the various segments of $w_j(\lambda)$ are separated by a (finite) number of breakpoints λ_k. In the next section we present an algorithm for obtaining the breakpoints and the efficient solutions.

3. An algorithm for generating the efficient solutions

A straightforward way of implementing the efficient solutions generation would be the following: take each $j \in N_f$ in turn, and, for that particular j, determine all the breakpoints in λ for the function $L_j(\bar{y}(j,\lambda),\lambda)$, thus providing the whole set of paths E_j. Such a procedure, however, would be computationally inefficient, since the informations obtained during the generation of the efficient solution set E_j for some $j \in N_f$ are not used in the generation of E_k for $k \neq j$.

As we now proceed to show, a good way of reducing the computational effort is to simultaneously generate all efficient solutions for all $j \in N_f$ by considering, for each λ, the whole shortest path tree with respect to the lengths $c_u + \lambda r_u$ on the arcs. It will be seen in Section 3.4 below that this appoach leads to a nice complexity result (namely a pseudopolynomial algorithm) which generalizes a previous result by Karp & Orlin [8] for a special class of parametric shortest path problems.

3.1. *Shortest path tree and efficient solutions*

Let $[P_A(\lambda)]$ denote the problem of calculating the shortest path tree $A(\lambda) = [N_1 \cup N_f, U(\lambda)]$ from s to all nodes of $N_1 \cup N_f$, with respect to the lengths $c_u + \lambda r_u$. The optimal solution $\bar{y}(j,\lambda)$ of problem $[P_j(\lambda)]$ is also the unique s-j path in this tree. Solving problem $[P_A(\lambda)]$ and not the problems $[P_j(\lambda)]$ one by one for each node $j \in N_f$ allows important computational savings.

It may be difficult to calculate explicitly the values of $\lambda_{\min}$ and $\lambda_{\max}$ (minimum and maximum values of λ for which there are no negative circuits in the graph G with respect to the costs $c_u + \lambda r_u$). During Algorithm 1, described later on, these calculations will be avoided by considering two phases:

(a) Phase I: solving problems $[P_A(\lambda)]$ for λ increasing from 0 to $\lambda_{\max}$.

(b) Phase II: solving problems $[P_A(\lambda)]$ for λ decreasing from 0 to $\lambda_{\min}$.

Each of these phases will terminate as soon as it has obtained a breakpoint $\bar{\lambda}$ such that the graph G_1 has a negative circuit with respect to the lengths $c_u + \bar{\lambda} r_u$.

3.2. *Reducing the set of efficient solutions*

We show here how to avoid generating efficient solutions that can not produce feasible solutions to problem [DCSP]. Let $r_{\min}(j)$ and $r_{\max}(j)$ be respectively the

lengths of the shortest path and of the longest path from $j \in N_f$ to t, taking r_u as the cost of arc $u \in U$:

$$r_{\min}(j) = \min_{z(j) \in X_{jt}(G_2)} \{r(z(j))\},$$

$$r_{\max}(j) = \max_{z(j) \in X_{jt}(G_2)} \{r(z(j))\},$$

where $r(z(j)) = \sum_{u \in U_2} r_u z_u(j)$.

A partial solution $y(j)$ might generate a feasible solution of [DCSP] only if

$$\underline{r} - r_{\max}(j) \leq r(y(j)) \leq \bar{r} - r_{\min}(j).$$

Thus, it is sufficient to consider just the efficient solutions satisfying this condition.

For $\lambda_1 < \lambda_2$ let $\bar{y}(j, \lambda_1)$ and $\bar{y}(j, \lambda_2)$ be the optimal solutions to the problems $P_j(\lambda_1)$ and $P_j(\lambda_2)$. Theorem 2 below is basic to the algorithm under study:

Theorem 2. *If* $\bar{y}(j, \lambda_1) \neq \bar{y}(j, \lambda_2)$, *then* $r(\bar{y}(j, \lambda_2)) < r(\bar{y}(j, \lambda_1))$.

Proof. If $\bar{y}(j, \lambda_1) \neq \bar{y}(j, \lambda_2)$ there exists at least one value $\bar{\lambda} \in (\lambda_1, \lambda_2]$ such that

$$c(\bar{y}(j, \lambda_1)) + \bar{\lambda} r(\bar{y}(j, \lambda_1)) = c(\bar{y}(j, \lambda_2)) + \bar{\lambda} r(\bar{y}(j, \lambda_2)).$$

Since $\bar{y}(j, \lambda_1)$ is an optimal solution to problem $P_j(\lambda_1)$, we have

$$c(\bar{y}(j, \lambda_1)) + \lambda_1 r(\bar{y}(j, \lambda_1)) < c(\bar{y}(j, \lambda_2)) + \lambda_1 r(\bar{y}(j, \lambda_2)).$$

Subtracting these equations we have

$$(\lambda_1 - \bar{\lambda}) r(\bar{y}(j, \lambda_1)) < (\lambda_1 - \bar{\lambda}) r(\bar{y}(j, \lambda_2))$$

and then $r(\bar{y}(j, \lambda_1)) > r(\bar{y}(j, \lambda_2))$ since $\lambda_1 < \bar{\lambda}$. □

Since an equivalent property holds for phase II, these results can be used in order to avoid obtaining efficient solutions that cannot produce feasible solutions to [DCSP]: it suffices to stop all calculations during phase I as soon as $r(\bar{y}(j, \lambda)) < \underline{r} - r_{\max}(j)$ and during phase II as soon as $r(\bar{y}(j, \lambda)) > \bar{r} - r_{\min}(j)$ for all $j \in N_f$.

3.3. *Finding the breakpoints in* λ

Remember that $A(\lambda) = [N_1 \cup N_f, U(\lambda)]$ is the tree defined as the solution of the problem $P_A(\lambda)$, having s as its root. We define $w(i, A(\lambda))$ as the path from s to i within $A(\lambda)$. This path is the current optimal path corresponding to the solution of the problem $[P_i(\lambda)]$. For each arc $u = (i, j)$ let

$$d_1(A(\lambda), u) = c(w(i, A(\lambda))) + c_u - c(w(j, A(\lambda))),$$

$$d_2(A(\lambda), u) = r(w(i, A(\lambda))) + r_u - r(w(j, A(\lambda))).$$

Assuming λ positive, we also define

$$\delta_u = \begin{cases} -d_1(A(\lambda), u)/d_2(A(\lambda), u), & \text{if } d_2(A(\lambda), u) < 0, \\ +\infty, & \text{otherwise.} \end{cases}$$

Theorem 3 below will be used to obtain the next breakpoint immediately following $\lambda \geq 0$:

Theorem 3. *Let* $\delta_{\bar{u}} = \min_{u \in U_1} \{\delta_u\}$ *and* $\bar{u} = (i, j)$. *Then, for any* $\gamma \in [\lambda, \delta_{\bar{u}}]$, $A(\lambda)$ *is the solution to the problem* $\mathrm{P}_A(\gamma)$.

Proof. Let $\pi(i) = c(w(i, A(\lambda))) + \lambda r(w(i, A(\lambda)))$. Then, for any $\gamma \in [\lambda, \delta_{\bar{u}}]$ and for any $u = (i, j) \in U_1$,

$$\pi(j) - \pi(i) \leq c_u + \gamma r_u.$$

Thus the values $\pi(i)$ satisfy the optimality conditions for the problem $\mathrm{P}_A(\gamma)$ and $A(\lambda)$ is its optimal solution. □

The breakpoint immediately following $\lambda \geq 0$ is thus $\bar{\lambda} = \delta_{\bar{u}}$. For $\gamma > \delta_{\bar{u}}$, $A(\lambda)$ is no longer the optimal solution to $[\mathrm{P}_A(\gamma)]$.

We denote by $P(A(\lambda), i)$ the set of nodes k of $N_1 \cup N_f$ such that the node i belongs to the path from s to k within $A(\lambda)$. Let $A_u(\lambda)$ be the graph obtained by introducing the arc $u = (i, j)$ in the tree $A(\lambda)$ and deleting from it the unique arc $v = (l, j) \in U(\lambda)$ having the same terminal endpoint as u. If $i \notin P(A(\lambda), j)$, then $A_u(\lambda)$ is a tree, otherwise a circuit is formed within $A_u(\lambda)$. Theorem 4 below enables us to obtain the solution of the new problem $[\mathrm{P}_A(\bar{\lambda})]$ without having to solve explicitly a shortest path problem:

Theorem 4. *Let* $\bar{\lambda} = \delta_{\bar{u}}$, $\delta_{\bar{u}}$ *being defined as in Theorem* 3. *If* $A_{\bar{u}}(\lambda)$ *is a tree, then* $A_{\bar{u}}(\lambda) = A(\delta_{\bar{u}})$ *is the solution of the problem* $[\mathrm{P}_A(\delta_{\bar{u}})]$.

Proof. We consider the tree $A_{\bar{u}}(\lambda)$ and we define

$$c(w(k, A_{\bar{u}}(\lambda))) = \begin{cases} c(w(k, A(\lambda))) + d_1(A(\lambda), \bar{u}), & \text{if } k \in P(A(\lambda), j), \\ c(w(k, A(\lambda))), & \text{otherwise,} \end{cases}$$

$$r(w(k, A_{\bar{u}}(\lambda))) = \begin{cases} r(w(k, A(\lambda))) + d_2(A(\lambda), \bar{u}), & \text{if } k \in P(A(\lambda), j), \\ r(w(k, A(\lambda))), & \text{otherwise.} \end{cases}$$

For all nodes $k \notin P(A(\lambda), j)$, $w(k, A_{\bar{u}}(\lambda)) = w(k, A(\lambda))$. For all nodes $k \in P(A(\lambda), j)$ we have

$$c(w(k, A_{\bar{u}}(\lambda))) + \delta_{\bar{u}} r(w(k, A_{\bar{u}}(\lambda))) = c(w(k, A(\lambda))) + \delta_{\bar{u}} r(w(k, A(\lambda))),$$

since $\delta_{\bar{u}} = -d_1(A(\lambda), \bar{u})/d_2(A(\lambda), \bar{u})$. By the definition of $\delta_{\bar{u}}$, we get

$$c(w(k, A(\lambda))) + \delta_{\bar{u}} r(w(k, A(\lambda))) = c(w(k, A(\delta_{\bar{u}}))) + \delta_{\bar{u}} r(w(k, A(\delta_{\bar{u}})))$$

and thus the tree $A_{\bar{u}}(\lambda)$ is the optimal solution to the problem $[\mathrm{P}_A(\delta_{\bar{u}})]$. □

This theorem makes it posible to obtain the solution of the problem $[P_A(\delta_{\bar{u}})]$ from the solution of the problem $[P_A(\lambda)]$ without recomputing explicitly the shortest path tree. Negative circuits are identified using Theorem 5 below:

Theorem 5. *If the graph $A_{\bar{u}}(\lambda)$ is not a tree, then it has a zero-length circuit with respect to the arc lengths $c_u + \delta_{\bar{u}} r_u$.*

Proof. If $A_{\bar{u}}(\lambda)$ is not a tree, then $i \in P(A(\lambda), j)$. Let d be the length of the circuit formed within $A_{\bar{u}}(\lambda)$. Then,

$$d = c(w(i, A(\lambda))) - c(w(j, A(\lambda))) + c_{\bar{u}} + \delta_{\bar{u}}[r(w(i, A(\lambda))) - r(w(j, A(\lambda))) + r_{\bar{u}}]$$

$$d = d_1(A(\lambda), \bar{u}) + \delta_{\bar{u}} d_2(A(\lambda), \bar{u}) = 0. \qquad \square$$

Finding a zero-length circuit within the graph $A_{\bar{u}}(\lambda)$ corresponds to solving a problem $[P_A(\delta_{\bar{u}})]$ with a zero-length circuit. In this case we can terminate generating the efficient solutions, since from this value of λ on, there will be always negative circuits.

3.4. *An algorithm for generating the efficient solutions*

We present below an algorithm for the generation of the efficient solutions. A balanced tree is used to store the values δ_u for all $u \in U$. This kind of data structure enables the insertion or the deletion of an element from it in $O(\log n)$ operations, where n is the number of elements already stored in the tree (for more details about balanced trees, see Knuth [9] and Aho, Hopcroft & Ullman [1]).

Algorithm 1 (generation of the efficient solutions)

Step 0. Let $\bar{\lambda} = 0$, $E_j = \emptyset$ and $k_j = 0$ $\forall j \in N_f$.
Solve the problem $[P_A(\bar{\lambda})]$ obtaining the tree $A(\bar{\lambda})$ as its solution.
For all $j \in N_f$, let $y^{k_j}(j) = \bar{y}(j, \bar{\lambda})$ and $E_j \leftarrow E_j \cup \{y^{k_j}(j)\}$.
For all $u \in U_1$ compute

$$\delta_u = \begin{cases} -d_1(A(\bar{\lambda}), u)/d_2(A(\bar{\lambda}), u), & \text{if } d_2(A(\bar{\lambda}), u) < 0, \\ +\infty, & \text{otherwise.} \end{cases}$$

Build the balanced tree with the values δ_u.

Step 1. Let $\lambda \leftarrow \bar{\lambda}$ and obtain from the balanced tree the breakpoint $\bar{\lambda}$ immediately following $\lambda > 0$, i.e.,

$$\bar{\lambda} = \delta_{\bar{u}} = \min_{u \in U_1} \{\delta_u\} \quad \text{and} \quad \bar{u} = (i, j).$$

If $\bar{\lambda} = +\infty$, then go to step 5.

Step 2. Let $A_{\bar{u}}(\lambda)$ be the graph obtained from the solution $A(\lambda)$ of the problem

solved previously by inserting $\bar{u}=(i,j)$ and deleting $v=(l,j)$ from $U(\lambda)$. If $A_{\bar{u}}(\lambda)$ is a tree, the solution $A(\bar{\lambda})=A_{\bar{u}}(\lambda)$ of the problem $[P_A(\bar{\lambda})]$ is such that

$$c(w(k,A(\bar{\lambda}))) = \begin{cases} c(w(k,A(\lambda))) + d_1(A(\lambda),\bar{u}), & \text{if } k \in P(A(\lambda),j), \\ c(w(k,A(\lambda))), & \text{otherwise,} \end{cases}$$

$$r(w(k,A(\bar{\lambda}))) = \begin{cases} r(w(k,A(\lambda))) + d_2(A(\lambda),\bar{u}), & \text{if } k \in P(A(\lambda),j), \\ r(w(k,A(\lambda))), & \text{otherwise,} \end{cases}$$

$$P(A(\bar{\lambda}),k) = P(A(\lambda),k), \quad k \neq i \text{ and } k \neq l$$

$$P(A(\bar{\lambda}),i) = P(A(\lambda),i) \cup P(A(\lambda),j), \quad \text{and}$$

$$P(A(\bar{\lambda}),l) = P(A(\lambda),l) - P(A(\lambda),j).$$

Step 3. If $A_{\bar{u}}(\lambda)$ is not a tree or if $r(\bar{y}(j,\bar{\lambda})) < \underline{r} - r_{\max}(j)\ \forall j \in N_f$, then go to step 5. Otherwise, for all $j \in N_f$ such that the conditions $\underline{r} - r_{\max}(j) \leq r(\bar{y}(j,\bar{\lambda})) \leq \bar{r} - r_{\min}(j)$ and $\bar{y}(j,\bar{\lambda}) \neq \bar{y}(j,\lambda)$ hold, set:

$$k_j \leftarrow k_j + 1, \quad y^{k_j}(j) = \bar{y}(j,\bar{\lambda}) \quad \text{and} \quad E_j \leftarrow E_j \cup \{y^{k_j}(j)\}.$$

Step 4. For all arcs $u=(a,b) \in U_1$ such that $a \in P(A(\bar{\lambda}),j)$ or $b \in P(A(\bar{\lambda}),j)$, recompute

$$\delta_u = \begin{cases} -d_1(A(\bar{\lambda}),u)/d_2(A(\bar{\lambda}),u), & \text{if } d_2(A(\bar{\lambda}),u) < 0, \\ +\infty, & \text{otherwise.} \end{cases}$$

Update the balanced tree and go back to step 1.

Step 5. Let $\bar{\lambda}=0$, $k_{\max}(j)=k_j$ and $k_j=0\ \forall j \in N_f$. Solve the problem $[P_A(\bar{\lambda})]$ obtaining the tree $A(\bar{\lambda})$ as its solution. For all $u \in U_1$ compute:

$$\delta_u = \begin{cases} -d_1(A(\bar{\lambda}),u)/d_2(A(\bar{\lambda}),u), & \text{if } d_2(A(\bar{\lambda}),u) > 0, \\ -\infty, & \text{otherwise.} \end{cases}$$

Build the balanced tree with the new values of δ_u.

Step 6. Let $\lambda \leftarrow \bar{\lambda}$ and obtain from the balanced tree the breakpoint $\bar{\lambda}$ immediately following $\lambda < 0$, with

$$\bar{\lambda} = \delta_{\bar{u}} = \max_{u \in U_1} \{\delta_u\} \quad \text{and} \quad \bar{u} = (i,j).$$

If $\bar{\lambda} = -\infty$, then go to step 10.

Step 7. Obtain the solution $A(\bar{\lambda}) = A_{\bar{u}}(\lambda)$ of the problem $[P_A(\bar{\lambda})]$ as in step 2.

Step 8. If $A_{\bar{u}}(\lambda)$ is not a tree or if $r(\bar{y}(j,\bar{\lambda})) > \bar{r} - r_{\min}(j)\ \forall j \in N_f$, then go to step 10. Otherwise, for all $j \in N_f$ such that the conditions $\underline{r} - r_{\max}(j) \leq r(\bar{y}(j,\bar{\lambda})) \leq \bar{r} - r_{\min}(j)$ and $\bar{y}(j,\bar{\lambda}) \neq \bar{y}(j,\lambda)$ hold, set[1]:

[1] Negative indices are used for the efficient solutions obtained from negative values of λ.

$$k_j \leftarrow k_j - 1, \qquad y^{k_j}(j) = \bar{y}(j, \bar{\lambda}) \quad \text{and} \quad E_j \leftarrow E_j \cup \{y^{k_j}(j)\}.$$

Step 9. For all arcs $u = (a, b) \in U_1$ such that $a \in P(A(\bar{\lambda}), j)$ or $b \in P(A(\bar{\lambda}), j)$, recompute

$$\delta_u = \begin{cases} -d_1(A(\bar{\lambda}), u)/d_2(A(\bar{\lambda}), u), & \text{if } d_2(A(\bar{\lambda}), u) > 0, \\ -\infty, & \text{otherwise.} \end{cases}$$

Update the balanced tree and go back to step 6.

Step 10. Let $k_{\min}(j) = k_j$ for all $j \in N_f$ and stop.

It is possible that at steps 1 and 6 there were several arcs $u \in U_1$ such that $\delta_u = \bar{\lambda}$, meaning that there would be more than one arc to introduce in the shortest path tree, which corresponds to degeneracy in the associated shortest path problem. Since we cannot introduce more than one arc in the shortest path tree at each iteration, the degeneracy will be solved by allowing the same value $\bar{\lambda}$ for more than one iteration: at each iteration the degeneracy is solved for only one node, introducing only one arc in the shortest path tree. The complexity of this algorithm is given by:

Theorem 6. *If all values r_u are positive, then Algorithm 1 is pseudopolynomial and runs in $O(Rm_1 \log n_1)$ time, where $R = \bar{r} - \min_{j \in N_i} \{r_{\min}(j)\}$, m_1 is the cardinality of U_1 and n_1 is the cardinality of $N_1 \cup N_f$.*

Proof. The value δ_u associated to each arc $u = (a, b)$ is computed each time the current optimal paths from s to a or from s to b are modified. Since the values $r(\bar{y}(a, \lambda))$ and $r(\bar{y}(b, \lambda))$ are monotonically decreasing with respect to λ during phase I (monotonically increasing during phase II), the maximum number of changes in the difference $r(\bar{y}(b, \lambda)) - r(\bar{y}(a, \lambda))$ (and so the maximum number of changes in the values of δ_u) is bounded by $2R$. Since there are m_1 arcs to be considered, there will be $2m_1 R$ calculations of values δ_u in the worst case. Since it is possible to insert, to delete or to modify each value in the balanced tree in $O(\log n_1)$ operations, the global complexity of Algorithm 1 is $O(Rm_1 \log n_1)$. □

Algorithm 1 is also an algorithm for solving the parametric shortest path problem where the arcs have lengths $c_u + \lambda r_u$. A particular case of this problem, where $r_u = 1$ for all arcs, has been studied by Karp & Orlin [8]. Though Algorithm 1 applies to a more general problem (where the values of r_u are arbitrary), it may be used to derive the results obtained by Karp and Orlin. In fact, in their particular case, the maximum number of distinct resource consumption values of the paths from s to each node of $N_1 \cup N_f$ is bounded by $(n_1 - 1)$, since the paths are required to be elementary, and the Algorithm 1 turns out to be polynomial, running in $O(n_1 m_1 \log n_1)$ operations.

4. Obtaining feasible solutions for the problem [DCSP]

We can now describe an algorithm that obtains feasible solutions for the problem [DCSP], using the sets E_j of efficient solutions generated by Algorithm 1.

In this algorithm, the sets of efficient solutions are explored in such a way as to minimize the total search time. At step 2 we choose a final solution $z(j)$ which will be combined with the various efficient solutions $y^k(j)$ obtained during the first phase of Algorithm 1. Remember that these efficient solutions are generated in an order such that we have always $r(y^k(j)) > r(y^{k+1}(j))$. Thus, as soon as a feasible solution of [DCSP] is obtained by combination with $z(j)$, we can stop the search of E_j, since all the other feasible solutions using $z(j)$ will have larger costs. The same criterion is used during the search of the efficient solutions generated during the second phase of Algorithm 1, and similar criteria are used to eliminate the efficient solutions that cannot belong to a feasible path.

Algorithm 2 (generation of the feasible solutions of problem [DCSP])

Step 0. Generate the sets E_j for all nodes $j \in N_f$ by using Algorithm 1:

$$E_j = \{y^k(j) \mid y^k(j) \text{ is an efficient solution and } k = k_{\min}(j), \ldots, 0, \ldots, k_{\max}(j)\}.$$

Let $\bar{N} = N_f$ and $UB = +\infty$.

Step 1. If $\bar{N} = \emptyset$ then go to step 9. Otherwise, choose $j \in \bar{N}$ and let

$$X = X_{jt}(G_2) \quad \text{and} \quad \bar{N} \leftarrow \bar{N} - \{j\}.$$

Step 2. If $X = \emptyset$, then go back to step 1. Otherwise, let $k = 0$ and choose $z(j) \in X$.

Step 3. Let $x = y^k(j) \oplus z(j)$. If $r \cdot x < \underline{r}$ then go to step 5. If $r \cdot x > \bar{r}$, then go to step 4. If $c \cdot x < UB$, then let $UB \leftarrow c \cdot x$ and $\bar{x} \leftarrow x$. Go to step 8.

Step 4. If $k < k_{max}(j)$, then set $k \leftarrow k+1$ and go back to step 3.

Step 5. If $k_{\min}(j) = 0$, then go to step 8. Otherwise, let $k = -1$.

Step 6. Let $x = y^k(j) \oplus z(j)$. If $r \cdot x > \bar{r}$ then go to step 8. If $r \cdot x < \underline{r}$, then go to step 7. If $c \cdot x < UB$, then let $UB \leftarrow c \cdot x$ and $\bar{x} \leftarrow x$. Go to step 8.

Step 7. If $k > k_{\min}(j)$, then set $k \leftarrow k-1$ and go back to step 6.

Step 8. Let $X \leftarrow X - \{z(j)\}$ and go back to step 2.

Step 9. Stop: $\bar{x}$ is the best feasible solution obtained.

5. Computational results and conclusions

Algorithm 2 was applied to doubly constrained shortest path problems obtained by relaxing 0–1 knapsack problems with an equality type constraint. In this case,

the graphs considered are sequential graphs associated to

$$a \cdot x \equiv b \text{ modulo}(m),$$

where $a=(a_j)$ is an integer valued n-vector, b and m are integers and $x=(x_j)$ is a n-vector of 0–1 bivalent variables. Ribeiro [13] and Minoux & Ribeiro [12] present a detailed study of this type of problem and of its use in the resolution of 0–1 knapsack problems with an equality constraint. Several criteria for obtaining the disconnecting set N_f for this kind of graph have been tested:

Criterion 1. Keep in N_2 the variables having the smallest lengths c_j: this criterion is oriented towards obtaining feasible paths with low costs (thus, with this criterion, better feasible solutions are usually obtained).

Criterion 2. Keep in N_2 the variables having the smallest coefficients r_j: this criterion is oriented towards minimizing the computing time necessary to obtain the feasible paths.

Criterion 3. Keep in N_2 the variables having the smallest products $c_j r_j$: this criterion is intended to represent a compromise between the first two criteria, and is oriented towards obtaining fairly good feasible paths in reduced computing times.

The graphs considered had up to 50 000 nodes and 100 000 arcs. The application of Algorithm 2 to 10 problems of this size leads to the results shown in Table 1.

These results show that the third criterion seems to be the most attractive one for practical purposes: the optimal solution is obtained in almost all the cases and the average number of efficient solutions generated (thus the computing times) is lower than what was obtained by the first criterion. In view of these results the algorithm can be considered to behave quite satisfactorily, since in practice it always obtained a feasible path, and this path indeed was, in the huge majority of cases, the exact optimal solution.

As a conclusion, we may say that the success of the approach presented here seems to confirm the relevance of the concept of an efficient solution for getting good approximate solutions to hard combinatorial optimization problems. Also we observe that the basic ideas presented here could easily be extended to quite a few

Table 1
Results of the application of Algorithm 2 to 10 problems. Average error = $|c \cdot \bar{x} - c \cdot x^*| / c \cdot x^*$, where $\bar{x}$ is the approximate solution and x^* is the optimal solution.

	Average number of efficient solutions generated for each node of N_f	Number of problems for which the optimal solution was obtained	Observed (average) relative error over 10 problems
Criterion 1	8.9	10/10	0.00%
Criterion 2	5.8	6/10	7.87%
Criterion 3	6.5	9/10	0.25%

other classes of discrete programming problems, at least those for which the associated parametric problem can be solved efficiently e.g., by a polynomial or pseudo-polynomial algorithm.

References

[1] A.V. Aho, J.E. Hopcroft and J.D. Ullman, Data Structures and Algorithms (Addison-Wesley, Reading, MA, 1983).

[2] J. Desrosiers, F. Soumis and M. Desrochers, Routing with time windows: synthesis, Research Report G-83-05, Ecole des Hautes Etudes Commerciales, Montréal (1983).

[3] H. Everett III, Generalized Lagrange multiplier method for solving problems of optimum allocation of resources, Operations Research 11 (1963) 399-417.

[4] M.R. Garey and D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness (W.H. Freeman, San Francisco, CA, 1979).

[5] J. Halpern and J. Priess, Shortest path with time constraints on movement and parking, Networks 4 (1974) 241-253.

[6] G.Y. Handler and I. Zang, A dual algorithm for the constrained shortest path problem, Networks 10 (1980) 293-310.

[7] H.C. Joksch, The shortest route problem with constraints, J. Math. Analysis Appl. 14 (1966) 191-197.

[8] R.M. Karp and J.B. Orlin, Parametric shortest path algorithms with an application to cyclic staffing, Discrete Appl. Math. 3 (1981) 37-46.

[9] D.E. Knuth, The Art of Computer Programming, Vol. 3: Sorting and Searching (Addison-Wesley, Reading, MA, 1973).

[10] M. Minoux, Plus courts chemins avec contraintes: algorithmes et applications, Ann. Télécommunications 30 (1975) 383-394.

[11] M. Minoux, Programmation Mathématique – Théorie et Algorithmes (Dunod, Paris, 1983). (English translation in preparation, Wiley, to appear.)

[12] M. Minoux and C. Ribeiro, A transformation of hard (equality constrained) knapsack problems, Oper. Res. Lett. 3 (1984) 211-214.

[13] C. Ribeiro, Algorithmes de recherche de plus courts chemins avec contraintes: etude théorique, implémentation et parallélisation, Docteur Ingénieur Dissertation, Ecole Nationale Supérieure des Télécommunications, Paris (1983).

[14] R. Saigal, A constrained shortest path problem, Operations Research 16 (1968) 205-209.